**暨南大学本科实验报告专用纸**

课程名称 算法分析与设计实验 成绩评定

实验项目名称 众数问题 指导教师 李展

实验项目编号 实验三 实验项目类型 综合性 实验地点

学生姓名 张印祺 学号 2018051948

学院 信息科学技术 系 计算机科学 专业 计算机科学与技术

实验时间 2020年3 月25 日下午～ 3月 25 日下午温度 ℃湿度

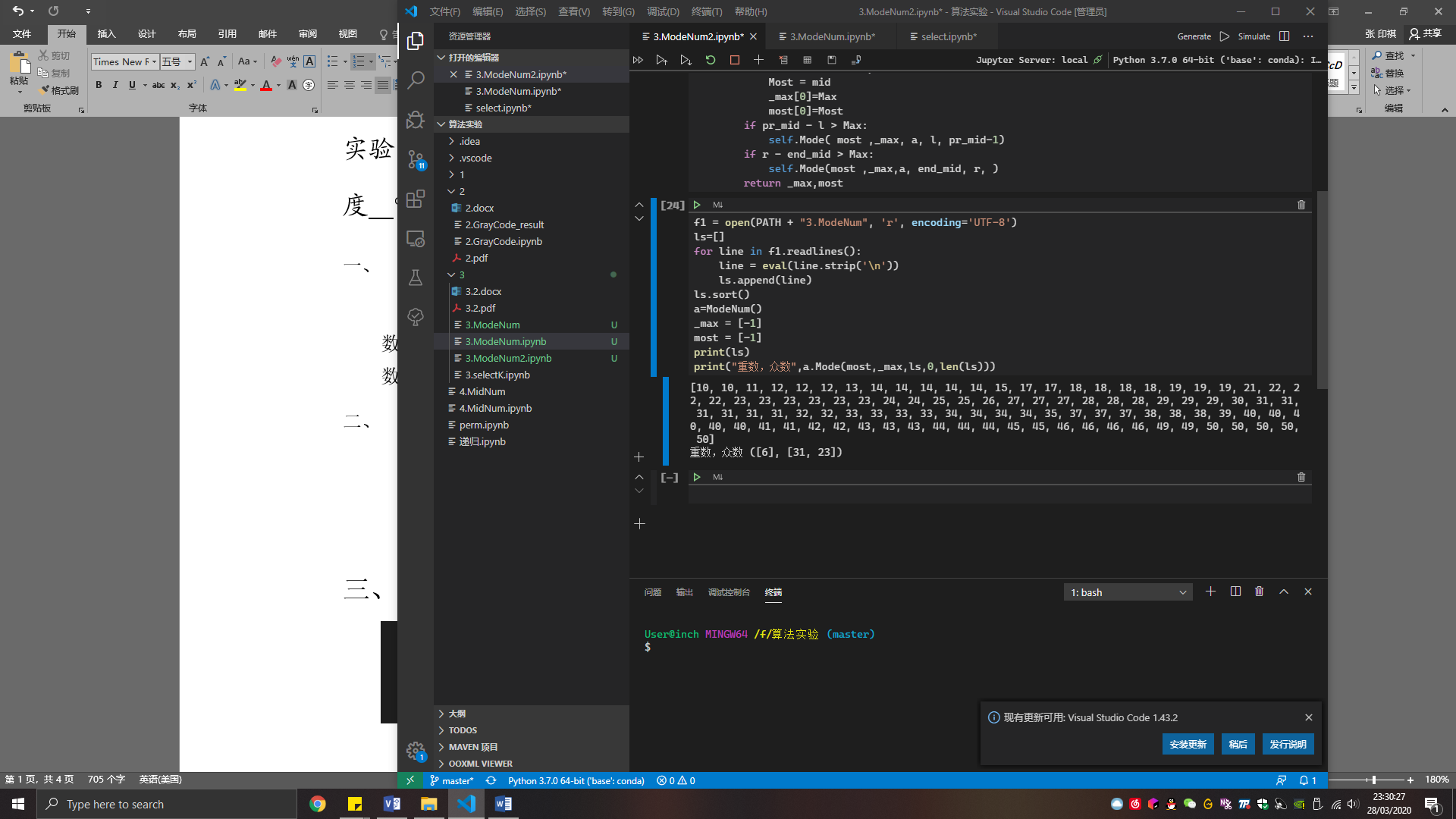
1. 问题描述

给定含有n个元素的多重集合S，每个元素在S中出现的次数称为该元素的重数。多重集S中重数的最大数称为众数。给定一个多重集合S，要求找出其中的众数以及其对应的重数。

1. 算法思路
2. 计算出我们的所选列表的中位数以及其长度将其存入临时变量 \_max
3. 使用分治法，若左侧长度大于\_max则在左侧继续选择中位数计算其长度，右侧同样的操作
4. 放弃长度小于max的未选序列可以大幅度降低时间复杂度。
5. 算法流程



1. 测试结果（左列为输入，右为输出）



1. 实验总结

本方法运用到了分治法，将元素进行分块至左右两侧列表大小均小于当前最大数为止，可以大幅度降低平均时间复杂度。

本方法的递推方程式为：，经过计算可得时间复杂度为O(n)。

但是本方法运用了分治的原理，他的最差时间复杂度并不低，如果我们采用hash表的方法可以达到平均时间复杂度与最差时间复杂度均为O(n)。其算法过程如下：

* + - 1. 新建一个dict={}
      2. 遍历list将list的值作为key，将出现次数作为dict[key]的值
      3. 遍历dict[key]输出最大数的key

Hash表方法代码如下：

import random as Rd

PATH = ".\\"

class ModeNum:

    def Creatdata():

        f = open(PATH + "3.ModeNum", 'w', encoding='UTF-8')

        for i in range(99):

            f.write(str(Rd.randint(10,50)) + '\n')

        f.close()

    def Select(dict):

        f1 = open(PATH + "3.ModeNum", 'r', encoding='UTF-8')

        list = f1.readlines()

        for i in range(len(list)):

            if list[i] in dict:

                dict[list[i]] += 1

            else:

                dict[list[i]] = 1

    def Search(dict, result, \_max):

        \_max = str(max(dict.keys(), key=(lambda x:dict[x])))

        result = ""

        for i in dict:

            if dict[i] == dict[\_max]:

                result += (str(eval(i)) + " ")

        print("众数：" + result + "重数：" + str(dict[\_max]) + "\n")

    def main(self, dict, result, \_max):

        ModeNum.Creatdata()

        ModeNum.Select(dict)

        ModeNum.Search(dict, result, \_max)

num = ModeNum()

num.dict = {}

num.result = ""

num.\_max=""

num.main(num.dict, num.result, num.\_max)

1. 源代码

import os

PATH = ".\\"

class ModeNum:

    def Mode(self, most ,\_max, a:list, l:int, r:int):

        Max=\_max[0]

        Mosy=most[0]

        mid = a[(l+r)//2]

        end\_mid = pr\_mid = a.index(mid)

        for i in range(pr\_mid, len(a)):      #记录最后结束位

            if a[i] == mid:

                end\_mid+=1

            else:break

        if Max == end\_mid - pr\_mid:

            most.append(mid)

        if Max < end\_mid - pr\_mid:

            most = [-1]

            Max = end\_mid - pr\_mid

            Most = mid

            \_max[0]=Max

            most[0]=Most

        if pr\_mid - l > Max:

            self.Mode( most ,\_max, a, l, pr\_mid-1)

        if r - end\_mid > Max:

            self.Mode(most ,\_max,a, end\_mid, r, )

        return \_max,most

f = open(PATH + "3.ModeNum", 'w', encoding='UTF-8')

for i in range(99):

    f.write(str(Rd.randint(10,50)) + '\n')

f.close()

f1 = open(PATH + "3.ModeNum", 'r', encoding='UTF-8')

ls=[]

for line in f1.readlines():

    line = eval(line.strip('\n'))

    ls.append(line)

ls.sort()

a=ModeNum()

\_max = [-1]

most = [-1]

print(ls)

print("重数，众数",a.Mode(most,\_max,ls,0,len(ls)))